Queen Mary
University of London

Main Examination period 2022/23

MTH6150: Numerical Computing in C and C++

Assignment date: Friday 7/4/2023
Submission deadline: Friday 12/5/2023 at 23:59 BST

The coursework is due by **Friday, 12th May 2021 at 23:59 BST**. Please submit a **report** (in `pdf` format) containing answers to **all** questions, complete with **written explanations** and printed tables or figures. Tables should contain a label for each column. Figures must contain a title, axis labels and legend. Please provide a brief explanation of any original algorithm used in the solution of the questions (if not discussed in lectures). **Code comments** should be used to briefly explain what your code is doing. You need to show that your program works using suitable examples. Build and run your code with any of the free IDEs discussed in class (such as CLion, Visual Studio, Xcode or an online compiler). The code produced to answer each question should be submitted aside with the `pdf` report, in a single `cpp` file, or multiple `cpp` files. **The code for each question should also be copied into your report**, so that it can be commented on when grading. Please include your name and student number on the first page of your report. Only material submitted through QMPlus will be accepted. Late submissions will be treated in accordance with current College regulations. **Plagiarism is an assessment offence and carries severe penalties.**

Coursework [100 marks]

Question 1. [20 marks] *Self-consistent iteration.*
Using a pocket calculator, one may notice that by applying the cosine key repeatedly to the value (in radians) one obtains a sequence of real numbers

$$\begin{aligned} x_1 &= \cos x_0 = 1. \\ x_2 &= \cos x_1 = 0.54030230586814 \\ x_3 &= \cos x_2 = 0.54030230586814 \\ x_4 &= \cos x_3 = 0.857553215846393 \\ &\vdots \\ x_{21} &= \cos x_{20} = 0.739184399771494 \\ &\vdots \end{aligned}$$

which tends to the value $x_\infty = 0.739085\ldots$, which is the point where the graphs of the function x and $\cos x$ intersect. The iteration can be written as

$$x_{n+1} = \cos x_n \text{ for } n = 0, 1, 2, \ldots \text{ with } x_0 = 1.$$

The limit x_∞ satisfies the transcendental equation

$$\cos x = x.$$

Write a `for` loop that performs the iteration $x_{n+1} = \cos x_n$ starting from the initial condition $x_0 = 0$ and stops when the absolute value of the difference $|x_{n+1} - x_n|$ between two consecutive iterations is less than a prescribed tolerance $\epsilon = 10^{-12}$. Use an `If/Break` statement to exit the loop when the condition is met. Print out the final value x_{n+1} to 16 digits of accuracy. In how many iterations did your loop converge? What is the final error in the above transcedental equation? (Hint: use the final value to compute and print out the difference $x - \cos x$.) **[20]**

Question 2. [20 marks] *Inner products.*

(a) Write a function that takes as input two Euclidean vectors $\vec{u} = \{u_1, u_2, ..., u_N\} \in \mathbb{R}^N$ and $\vec{v} = \{v_1, v_2, ..., v_N\} \in \mathbb{R}^N$ (of type `valarray<long double>`) and returns their inner product (also known as Hadamard product)

$$\vec{u} \cdot \vec{v} = \sum_{i=1}^{N} u_i v_i \tag{1}$$

as a `long double` number. Your function may use `(u*v).sum()` to compute the dot product of the `valarrays` `u` and `v`. Create a constant `valarray<long double>` equal to `u={0.1,0.1,...,0.1}` with $N = 10^6$ elements. Demonstrate that your program works by computing the dot product $\vec{u} \cdot \vec{u}$ for this constant vector. Display the difference $\vec{u} \cdot \vec{u} - 10^4$ on the screen. **[5]**

(b) Repeat Question 2a using Kahan compensated summation to compute the sum. **[5]**

(c) Write code for a function object that has a member variable m of type `int`, a suitable constructor, and a member function of the form

```
double operator()(const valarray<double> u) const {
   ...
}
```

which returns the weighted norm

$$\ell_m(\vec{v}) = \sqrt[m]{\sum_{i=0}^{N} |v_i|^m} \tag{2}$$

Use this function object to calculate the norm $\ell_2(\vec{u})$ for the vector in Question 2a. Does the quantity $\ell_2(\vec{u})^2$ equal the inner product $\vec{u} \cdot \vec{u}$ that you obtained above? [*Note*: half marks awarded if you use a regular function instead of a function object.] **[10]**

Question 3. [20 marks] *Finite differences*.

(a) Write a C++ program that uses finite difference methods to numerically evaluate the first derivative of a function $f(x)$ whose values on a fixed grid of points are specified $f(x_i),\ i = 0, 1, ..., N$. Your code should use three instances of a `valarray<long double>` to store the values of x_i, $f(x_i)$ and $f'(x_i)$. Assume the grid-points are located at $x_i = a + i\,\Delta x$ with $\Delta x = (b-a)/N$. on the interval $x \in [a, b]$ and use 2nd order finite differencing to compute an approximation for $f'(x_i)$:

$$\begin{aligned} f'(x_0) &= \frac{-3f(x_0) + 4f(x_1) - f(x_2)}{2\Delta x} + O(\Delta x^2) \quad \text{for} \quad i = 0 \\ f'(x_i) &= \frac{f(x_{i+1}) - f(x_{i-1})}{2\Delta x} + O(\Delta x^2) \quad \text{for} \quad i = 1, 2, ..., N-1 \\ f'(x_N) &= \frac{f(x_{N-2}) - 4f(x_{N-1}) + 3f(x_N)}{2\Delta x} + O(\Delta x^2) \quad \text{for} \quad i = N \end{aligned}$$

Demonstrate that your program works by evaluating the derivatives of a known function, $f(x) = \sin 3x$, with $N + 1 = 32$ points on the interval $x \in [a, b] = [-1, 1]$. Compute the difference between your numerical derivatives and the known analytical ones:

$$e_i = f'_{\text{numerical}}(x_i) - f'_{\text{analytical}}(x_i)$$

at each grid-point. Output the values e_i of this `valarray<long double>` on the screen and tabulate (or plot) them in your report. **[10]**

(b) For the same choice of $f(x)$, demonstrate 2nd-order convergence, by showing that, as N increases, the mean error $\langle e \rangle$ decreases proportionally to $\Delta x^2 \propto N^{-2}$. You may do so by tabulating the quantity $N^2\langle e \rangle$ for different values of N (e.g. $N + 1 = 16,\ 32,\ 64,\ 128$) and checking if this quantity is roughly constant. Alternatively, you may plot $\log\langle e \rangle$ vs. $\log N$ and check if the dependence is linear and if the slope is -2.
Here, the mean error $\langle e \rangle$ is defined by

$$\langle e \rangle = \frac{1}{N+1}\sum_{i=0}^{N} |e_i| = \frac{1}{N+1}\ell_1(\vec{e}).$$

You may use your code from Question 3c to compute the ℓ_1 norm. **[10]**

Question 4. [20 marks] *Numerical integration.*
We wish to compute the definite integral

$$I = \int_a^b \sin\left(\frac{1}{x+\frac{1}{2}}\right) dx$$

numerically for $a = 0$, $b = 10$ and compare to the exact result, $I_{\text{exact}} = 2.74324739415100920$.

(a) Use the composite trapezium rule

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{N} w_i f_i, \quad w_i = \begin{cases} \Delta x/2, & i = 0 \text{ or } i = N \\ \Delta x & 1 \le i \le N-1 \end{cases}, \quad \Delta x = \frac{b-a}{N},$$

to compute the integral I, using $N + 1 = 128$ equidistant points in $x \in [a, b]$. Use three instances of a `valarray<long double>` to store the values of the gridpoints x_i, function values $f_i = f(x_i)$ and weights w_i. [Hint: you may use the function from Question 2a to compute the dot product of the `valarrays` w_i and f_i.] Output to the screen (and list in your report) your numerical result $I_{\text{trapezium}}$ and the difference $I_{\text{trapezium}} - I_{\text{exact}}$. **[5]**

(b) Use the composite Hermite rule

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{N} w_i f_i + \frac{\Delta x^2}{12}[f'(a) - f'(b)]$$

with the derivatives $f'(x)$ at $x = a$ and $x = b$ evaluated analytically (and the weights w_i identical to those given above for the trapezium rule), to compute the integral I, using $N + 1 = 128$ equidistant points in $x \in [0, 1]$. Output to the screen (and list in your report) your numerical result I_{Hermite} and the difference $I_{\text{Hermite}} - I_{\text{exact}}$. **[5]**

(c) Use the Clenshaw-Curtis quadrature rule

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{N} w_i f_i, \quad w_i = \frac{b-a}{2} * \begin{cases} \frac{1}{N^2}, & i = 0 \text{ or } i = N \\ \frac{2}{N}\left(1 - \sum_{k=1}^{(N-1)/2} \frac{2\cos(2k\theta_i)}{4k^2-1}\right) & 1 \le i \le N-1 \end{cases},$$

on a grid of $N + 1 = 128$ points $x_i = [(a + b) - (b - a)\cos\theta_i]/2$, where $\theta_i = i\pi/N$, $i = 0, 1, ..., N$ to compute the integral I. [Hint: First compute the values of θ_i, x_i, $f_i = f(x_i)$ and w_i and store them as `valarrays`. Then, you may use the function from Question 2a to compute the dot product of the `valarrays` w_i and f_i.] Output to the screen (and list in your report) your numerical result $I_{\text{ClenshawCurtis}}$ and the difference $I_{\text{ClenshawCurtis}} - I_{\text{exact}}$. **[5]**

(d) Compute the integral I using a *Mean Value* Monte Carlo method with $N = 1000$, $N = 10000$ and $N = 100000$ samples. Output to the screen (and list in your report) your numerical results $I_{\text{MonteCarlo}}$ and the difference $I_{\text{MonteCarlo}} - I_{\text{exact}}$ for each N. **[5]**

Question 5. [20 marks] *Stellar equilibrium.*
Consider the generalized Lane-Emden equation, in the form of the initial value problem

$$\begin{cases} h''(x) + \frac{2}{x}h'(x) + h(x) = 0 \\ h(0) = 1,\ h'(0) = 0 \end{cases} \tag{3}$$

This equation describes the structure of a self-gravitating planet or stellar object with mass density $\rho(h) = h$, where h is the specific enthalpy. Setting $m(x) := -x^2 h'(x)$, the above 2nd-order differential equation can be reduced to a system of two 1st-order differential equations for $h(x)$ and $m(x)$:

$$\begin{cases} h'(x) = \begin{cases} 0 & x = 0 \\ -\frac{1}{x^2}m(x) & x > 0 \end{cases} \\ m'(x) = h(x)\, x^2 \\ h(0) = 1 \\ m(0) = 0 \end{cases} \tag{4}$$

The above equation appears singular at $x = 0$, but one can use a Taylor expansion about the origin to show that $m(x) \simeq \frac{2}{3}x^3 + \mathcal{O}(x^5)$ for small x, so that $h'(0) = 0$.

(a) Declare a function

```
valarray<long double> F(const long double t, const
valarray<long double>& u) {
   ...
}
```

that takes the radius x and a valarray with elements $\vec{u} = \{h, m\}$ as arguments and returns a valarray with components $\vec{f} = \{h', m'\}$ as output, with h', m' given by Eq. (4) above. **[5]**

(b) Solve the above 1st-order system numerically, with a 4th-order Runge-Kutta method

```
long double RK4(const double t, const double dt, const
valarray<long double> &, long double f(const long double,
valarray<long double> &) {
   ...
}
```

using $N + 1 = 151$ equidistant points in $x \in [0, \pi]$.

Declare a valarray of valarrays:

```
valarray<valarray<long double>> u
```

and use it to store $\vec{u}(x_i) = \{h_i, m_i\} = \{h(x_i), m(x_i)\}$ at each grid-point x_i.

Output (only) the values $\{x_0, x_{10}, x_{20}, ..., x_N\}$ and $\{h(x_0), h(x_{10}), h(x_{20}), ..., h(x_N)\}$ to the screen and tabulate them in your report. **[5]**

(c) Compute the difference $e(x) = h_{\text{numerical}}(x) - h_{\text{exact}}(x)$ between your numerical solution $h_{\text{numerical}}(x)$ and the exact solution

$$h_{\text{exact}}(x) = \text{sinc}\, x = \begin{cases} \frac{\sin x}{x} & x \neq 0 \\ 1 & x = 0 \end{cases}$$

Output the error values $e(x_0), e(x_{10}), e(x_{20}), ..., e(x_N)$ to the screen and tabulate them in your report. Comment on whether the error is what you expected and why. What is the radius $x = R$ of the star, where $h_{\text{exact}}(x) = 0$? [Hint: use your answer from Question 1].

Remark: it is more convenient to display $\{x_i\}$, $\{h(x_i)\}$, $\{e(x_i)\}$ all at once, that is, answer Questions 5b and 5c in the same loop. **[5]**

(d) Compute the error norms:

$$l_1(\vec{e}) = \sum_{i=0}^{N} |e_i|, \quad l_2(\vec{e}) = \sqrt{\sum_{i=0}^{N} |e_i|^2}$$

where $e_i = e(x_i)$ is the error at each grid-point. [Hint: you may use your C++ implementation of Eq. (2) from Question 2 to compute the norms.] Comment on whether the error norms are what you expected. **[5]**

End of Paper.